Problem 2. ePath

Input file:	`epath.in`
Output file:	`epath.out`
Time limit:	15 секунд
Memory limit:	64 megabytes

Известная берляндская оффшорная фирма «ePath» получила заказ от крупной логистической компании. Разработчикам «ePath» надо написать программу, автоматизирующую нахождение кратчайшего маршрута из точки A в точку B. Было решено считать землю плоской, а все препятствия попарно непересекающимися отрезками $[A_1B_1], [A_2B_2], \ldots, [A_N, B_N]$.

Программисты компании быстро сообразили, что для решения задачи надо обратиться к профессору Л. Профессор заверил, что это идейно несложная задача на кратчайшие пути во взвешенных графах. Также он заметил, что в процессе построения графа могут возникнуть хитрые проблемы с точностью, на что представители заказчика заверили, что входные данные всегда таковы, что все вопросы, касательные точности, можно решить, используя вычисления в типе double (8 байтовый вещественный тип).

Программисты справились с реализацией алгоритма Дейкстры, а вот с построением графа у них возникли проблемы. Заказчики недовольны алгоритмом за $O(N^3)$ и согласны исключительно на быструю реализацию более эффективного решения.

Назовем точку P видимой из точки Q, если не существует такого препятствия $[A_i, B_i]$, что отрезки $[P, Q]$ и $[A_i, B_i]$ пересекаются, и точки P, Q не лежат на прямой A_iB_i, но лежат по разные стороны от нее. То же самое должно быть верно для точек A_i, B_i и прямой PQ.

Занумеруем точки таким образом, что $P_{2i-1} = A_i$, а $P_{2i} = B_i$. Определим неориентированный граф, вершинами которого являются точки $P_1, P_2, \ldots, P_{2N}$, а ребро между P_i и P_j существует, если P_i видна из P_j.

Ваша задача написать часть проекта, строящую матрицу смежности искомого графа.

Input

В первой строке входного файла записано целое число N ($1 \le N \le 700$), где N — количество отрезков. Далее записаны сами отрезки парами координат точек A_i, B_i. Координаты — целые числа, по модулю не превосходят 10^6. Все отрезки попарно не пересекаются Гарантируется, что для любой точки P_k и прямой A_iB_i либо точка P_k лежит на этой прямой, либо расстояние от нее до прямой не менее 0.001.

Output

Выведите N строк по N символов в каждой. Поставьте символ $\widehat{0}$ в позицию (i, j) в случае отсутствия ребра (P_i, P_j) и $\widehat{1}$ в противном случае.

Example

epath.in	epath.out
3 0 0 0 10 10 0 10 10 20 0 20 10	111110 111101 111111 111111 101111 011111

Problem 3. Конь и кубик Рубика

Input file:	`knight.in`
Output file:	`knight.out`
Time limit:	5 секунд
Memory limit:	64 megabytes

ВНИМАНИЕ: Задача является «аппроксимационной»: гарантируется, что авторское решение работает за отведённое время на всех проверенных автором тестах (в том числе, разумеется, и на предлагаемых), но не доказано, что на произвольном тесте авторское решение уложится в отведённое время.

Вдохновившись примером слона и ладьи из задачи H, конь тоже решил побродить по поверхности куба. Поскольку он был один, то задачу перед собой поставил более привычную — обойти поверхность куба, побывав на каждой клетке ровно один раз. Но по ошибке вместо обычного куба он попал на кубик Рубика (размером $3 \times 3 \times 3$), причем как раз в то время, когда с этим кубиком кто-то играл. В результате ходы коня и повороты грани кубика чередовались, и если конь оказывался на слое куба, который поворачивали, он двигался вместе с этим слоем. «Следы» коня (клетки, на которых он успел побывать) тоже перемещались. Конь ходит «зигзагом»: перемещается на одну клетку, поворачивает налево или направо, перемещается на одну клетку, потом поворачивает направо или налево соответственно и ещё раз перемещается на одну клетку.

Требуется найти какой-нибудь маршрут коня (длиной 53 хода), который обходит все клетки поверхности кубика.

Input

Длина ребра куба - 6 см, он расположен так, что задается неравенствами $0 \leqslant X \leqslant 6, 0 \leqslant Y \leqslant 6, 0 \leqslant Z \leqslant 6$ Грани куба $X = 0, Y = 0, Z = 0, X = 6, Y = 6, Z = 6$ обозначим буквами A, B, C, D, E, F соответственно. Повороты граней задаются буквой и направлением: $A+$ — поворот грани A по часовой стрелке, $B-$ — поворот грани B против часовой стрелки, $C2$ - поворот грани C на 180 гр. Система координат X, Y, Z «правая», т.е. если ось X направлена вправо, а Y — вдаль, то ось Z направлена вверх.

Входной файл содержит строку, содержащую три целых числа — координаты центра клетки, на которой стоит конь, и одну или несколько строк, содержащих не менее 104 непробельных символов — описание 52 поворотов граней кубика. Считается, что первый ход конь делает до поворота первой грани.

Output

В выходной файл требуется вывести 159 чисел — координаты центров клеток, на которые ходит конь. Координаты стартовой клетки в поток не выводятся.

Example

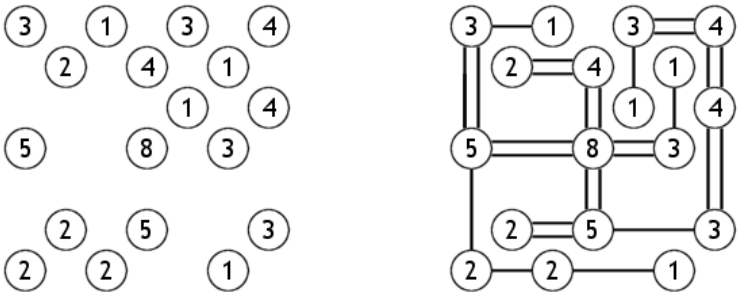
knight.in	knight.out
0 1 1 A+ A+ A+ A+ A+ A+ A+ A+ A+ A+ A+ A+ A+ A+ A+ A+ A+ A+ A+ A+ A+ A+ A+ A+ A+ A+ A+ A+ A+ A+ A+ A+ A+ A+ A+ A+ A+ A+ A+ A+ A+ A+ A+ A+ A+ A+ A+ A+ A+ A+ A+ A+	0 5 3 1 6 5 1 0 3 0 5 3 1 0 5 1 0 3 1 0 1 0 3 3 1 5 6 0 1 1 3 6 3 5 5 6 3 1 6 5 0 3 3 1 0 0 1 1 1 1 0 0 3 1 1 5 0 1 3 6 0 3 1 0 1 5 1 0 5 3 0 3 5 1 6 3 5 6 5 6 3 3 5 0 5 1 0 6 3 3 5 5 0 6 1 1 6 3 5 5 0 5 3 0 1 6 1 3 6 5 1 3 3 0 5 0 1 1 0 3 1 0 1 5 3 0 3 6 1 5 6 5 3 3 6 6 5 5 5 6 1 6 3 1 6 1 5 3 0 5 5 3 6 6 5 3 3 6 5

Problem 5. Bridges

Input file:	`bridges.in`
Output file:	`bridges.out`
Time limit:	2 seconds
Memory limit:	64 megabytes

Bridges is a logic puzzle where the challenge is to connect a number of islands on a grid using horizontal and vertical bridges so that the following conditions hold:

- The number of bridges connected to an island matches the given number for that island
- At most 2 bridges connect the same two islands
- Bridges may not cross each other, or other islands
- It is possible to go from one island to all other islands



The figure above to the left shows the puzzle layout, and the picture above to the right the unique solution. No two islands will lie next to each other, horizontally or vertically. Only puzzles that have one unique solution are valid Bridges puzzles.

Input

The input will contain several valid Bridges puzzles. Each puzzle starts with a line containing two integers, the width ($1 \leqslant w \leqslant 13$) and height ($1 \leqslant h \leqslant 22$) of the grid. The following h lines with w characters each describe the grid. Empty grid squares will be represented by a period, '`.`', and islands by a digit between '`1`' and '`8`', inclusive (the exact number of bridges this island should have). A blank line separates each puzzle. The last puzzle will have width and height 0, and should not be processed.

Output

For each puzzle, output the same grid as the input but with the bridges in place. The characters '`-`' and '`=`' should be used for single and double horizontal bridges, respectively, and the characters '`|`' and '`"`' for single and double vertical bridges, respectively. Separate each puzzle with a blank line.

Example

bridges.in	bridges.out
7 7 3.1.3.4 .2.4.1. ....1.4 5..8.3. .... .2.5..3 2.2..1. 9 5 2.2.2.2.2 .... 2.......2 .... 2.2.2.2.2 0 0	3-1.3=4 "2=4|1" ".."1|4 5==8=3" |..".." |2=5-3 2-2-1. 2-2-2-2-2 |.......| 2.......2 |.......| 2-2-2-2-2

Problem 7. Обмен колдовским опытом

Input file:	`natural.in`
Output file:	`natural.out`
Time limit:	2 секунды
Memory limit:	64 megabytes

В порядке изучения колдовских традиций различных народов делегация Хогвартса во главе с профессором Флитвиком устроила экспедицию в некоторую отдалённую страну. Но на пути экспедиции возникла проблема: если жители городов еще умеют пользоваться деньгами, то племена, контролирующие торговые пути между городами, не слышали, что такое сикли или галлеоны и предпочитают оплату натурой. Иными словами, владелец каравана, следующего из одного города в другой, должен отдать в качестве платы за проход определенное количество шкур, руды, огненной воды или других предметов, нужных племени. К счастью, эти предметы можно купить за золото в городах, через которые проходит караван.

Вам необходимо провести караван экспедиции из одного города в другой, затратив на плату за проход между городами минимальное количество золота. Сможете ли Вы сделать это?

Input

Первая строка содержит параметры задачи: значения N (количество городов в стране, пронумерованных, начиная с единицы), M (количество торговых путей) и K (количество предметов, используемых в качестве платы за проезд). Ограничения на параметры задачи: $2 \le N \le 1000$, $1 \le M \le 10000$, $1 \le K \le 10$. Каждая из последующих N строк входного файла описывает цену предметов, используемых в качестве платы за проход, в этом городе. Строка содержит K неотрицательных чисел, не превосходящих 1000 — цену единицы каждого предмета. Если какое-либо число равно 0, соответствующий предмет нельзя купить в этом городе, в противном случае количество предметов, которые можно купить, не ограничивается. Наконец, каждая из последующих M строк файла описывает один торговый путь и содержит $K+2$ неотрицательных числа: номер города - пункта отправления, номер города — пункта назначения и количество предметов каждого вида (не превосходящее 1000), которые необходимо отдать в качестве платы за проход по этому пути. Если количество какого-либо предмета в описании пути равно 0, соответствующий предмет нельзя использовать в качестве оплаты прохода по этому пути. Если эта величина равна 0 для всех возможных предметов, плата за проход по данному пути не требуется. Все пути — односторонние, но из одного города в другой может существовать несколько различных путей. Вам необходимо провести караван зкспедиции из города 1 в город N. Выбрав один из путей, соединяющих два города, Вы не можете в дальнейшем свернуть с него до прихода в пункт назначения. При оплате за проход по любому пути Вы можете использовать только один из предметов, необходимых племени (на Ваш выбор). В начале пути никаких предметов, используемых в качестве оплаты, у экспедиции нет, но Вы можете сделать первую закупку в городе 1. Все числа - целые и отделяются друг от друга одним или несколькими пробелами.

Output

Единственная строка выходного файла должна содержать минимальное количество золота, которое необходимо истратить в качестве платы за проход. Если задача не имеет решения, вы должны вывести в выходной файл значение -1.

Пояснение к примеру: Обозначим предметы оплаты через A и B. Вы должны купить в городе 1 четыре предмета A, перейти в город 2, купить там пять предметов B и вернуться в город 1. Теперь можно перейти в город 3, расплатившись за этот проход предметами B.

Example

natural.in	natural.out
3 4 2 1 2 2 1 0 1 1 2 2 0 2 1 2 0 2 3 15 10 1 3 10 5	9

Problem 11. Таинственный замок

Input file:	`input.txt`
Output file:	`output.txt`
Time limit:	20 seconds
Memory limit:	64 megabytes

Дон Рипат, верный агент дона Руматы, сообщил ему про предстоящую тайную встречу дона Рэбы и предводителя разбойников Ваги Колеса. Судя по всему, обсуждаться должно было что-то достаточно важное. «Вероятно, стоит попробовать подслушать разговор» — решил Антон. Но на двери, ведущей в покои дона Рэбы, висел замок странной конструкции. Это был один из знаменитых ируканских замков. Один такой замОк стоил целого зАмка. На замке была нарисована полоска из N равных квадратиков, в открытом состоянии последовательно меняющих свой цвет от белого к чёрному. При закрывании квадратики случайно расставлялись. Можно было взять несколько подряд идущих квадратиков и развернуть весь этот прямоугольник на 180 градусов — механизм замка позволял это сделать для любого числа квадратиков. Замок открывался приведением полоски в исходное состояние. Сразу же после того, как информация поступила на базу, Вам была поставлена задача написать программу, которая бы открывала подобный замок за минимальное число разворотов.

Input

Во входном файле в первой строке задано число N ($1 < N < 12$) — количество квадратиков в полоске замка. В следующей строке задана начальная конфигурация замка. N различных чисел от 1 до N, соответствующие определенному оттенку серого цвета, записаны через пробел друг за другом.

Output

В первую строку выходного файла вывести число M — минимальное количество разворотов, с помощью которых можно открыть замок. Во вторую строку вывести исходную конфигурацию замка. Далее должны следовать M строк, каждая из которых должна содержать очередную конфигурацию замка, полученную из предыдущей с помощью одного разворота. Таким образом, в последней строке выходного файла всегда должна быть конфигурация, соответствующая исходному состоянию замка.

Example

input.txt	output.txt
5 3 5 2 4 1 3 5 2 4 1 3 2 5 4 1 3 2 1 4 5 1 2 3 4 5	3

Problem 13. Краны

Input file:	`faucets.in`
Output file:	`faucets.out`
Time limit:	2 секунды
Memory limit:	64 megabytes

Теория Большого Взрыва — это подарок химиков физикам.

лекция по экспериментальной химии

В алхимической лаборатории, которую Скив и Ааз нашли в трактире Иштвана, над доской длиной X миллиметров были расположены n кранов. Все краны расположены на высоте Y миллиметров, в позициях X_1, X_2, $\ldots X_n$ миллиметров от начала доски ($0 \le X_1 < X_2 < \ldots < X_n \le X$), координаты нумеруются слева направо. Краны были закрыты неплотно, и поэтому из них иногда капало. Капли из каждого крана падают с периодичностью P_i секунд ($0 < P_i < 100$) с ускорением свободного падения, равным в измерении Пент (в котором и происходит дело) $g = 10$ м/с^2 (начальная скорость каждой капли равна 0). Но капает не вода, а разные растворы. Скив, которому его наставник Ааз поручил присматривать за лабораторией, заметил, что при достижении доски капли ведут себя очень странным образом. А именно, они начинают ползти по доске влево или вправо (в зависимости от крана) с заданной скоростью $d_i > 0$ мм/с. Каждая ползущая капля оставляет за собой осадок высотой h_i миллиметров, начиная с точки ее падения. Как только встречаются две или более капель разных растворов, все встретившиеся капли взрываются (при этом в точке взрыва осадок не остается). Если какая-то капля доползает до края доски, то она срывается вниз и больше не появляется.

Всё это было очень интересно. И тут Скив услышал, что в дверь трактира кто-то постучал. Значит, и досмотреть, чем всё закончится, не удастся. «Интересно, что будет с доской через T секунд», - подумал Скив.

Input

Во входном файле задано не более 5 наборов входных данных. В первой строке каждого набора заданы числа n, X, Y, T ($1 \le n \le 10$, $0 \le X, Y \le 10^9$, $0 \le T \le 100$). Далее следуют n строк, описывающих краны. Каждый кран описывается шестью полями: X_i d_i h_i P_i R_i c_i, где $0 \le h_i \le 100$, $d_i \le 100$, c_i соответствует направлению движения капли, и равно `+`, если капля ползет вправо (увеличивает x-координату) и `-` в противном случае. R_i ($0 \le R_i < P_i$) обозначает, что первая капля упадет из крана на R_i-й секунде. Все числа во входном файле целые.

Входной файл завершается фиктивным набором с $n = X = Y = T = 0$.

Output

Для каждого набора входных данных выведите состояние доски через T секунд (учитывая события, произошедшие ровно в момент времени T). Сначала в отдельной строке выведите информацию о номере набора. Далее выведите высоту участков доски в виде одного или нескольких интервалов, объединение которых составляет всю доску, в порядке их следования. Круглая скобка на границе интервала обозначает, что соответствующая точка в него не включается, квадратная — что включается. Гарантируется, что в момент времени T высота всех участков доски строго меньше Y.

Разделяйте выводы для различных наборов пустой строкой. Следуйте формату, показанному в примере, максимально точно. Вещественные числа выводите с максимально возможной точностью!

Example

faucets.in
2 5 5000 5 1 1 5 2 1 + 4 2 3 4 0 - 0 0 0 0
faucets.out
Test case 1: on [0; 1): height 0 on [1; 1.333333333333333): height 10 on [1.333333333333333, 1.333333333333333]: height 5 on [1.333333333333333, 2]: height 8 on (2, 4) height: 3 on [4, 4] height: 6 on (4, 5] height: 0

Problem 17. Погоня на Лимбе

Input file:	`mirror.in`
Output file:	`mirror.out`
Time limit:	2 секунды
Memory limit:	64 megabytes

Свет мой, зеркальце, ска...

Медуза Горгона

— А вот это уже интересно, — заметил Ааз, смотря куда-то вниз. Скив посмотрел туда же, но ничего особенного не увидел. Какие-то осколки... всего n выпуклых фигур. Скив присмотрелся и заметил, что граница осколков составлена из отрезков и дуг окружностей, более того, граница каждого осколка имеет хотя бы одну дугу. Всё равно непонятно, что в этом нашёл Ааз...

— Что там, босс? — спросил Скива Гвидо.

— Просто осколки зеркала, — ответил Скив.

— И откуда же им взяться в Лимбо? — Ааз перешёл к своей обычной манере.

Скив понял, про что говорил его учитель. Лимбо — это измерение, заселённое вампирами. Вампиры, скажем прямо, не большие охотники пользоваться зеркалами. А значит, зеркало могло принадлежать тем, кого они искали.

Гвидо вспомнил, что вроде бы действительно, что-то типа круглого зеркала он у мошенников видел.

— Не факт, что разбившееся зеркало круглое, — скептически заметил Скив.

Требуется определить, можно ли сложить осколки в один круг.

Input

Во входном файле задано не более десяти наборов входных данных. Количество наборов T задано в первой строке. Описание каждого набора начинается с числа выпуклых фигур n ($1 \le n \le 8$). Описание каждой фигуры начинается со строки, содержащей количество ограничивающих ее примитивов k_i ($1 \le k_i \le 8$). После этого идет k_i строк, описывающих примитивы. Граница рисуется, начиная с точки $(0, 0)$, и заканчивая в этой же точке, в порядке обхода против часовой стрелки. Все отрезки и дуги задаются тремя числами — x_i y_i k_i, где x_i и y_i — точки окончания соответствующего примитива (отрезка или дуги), а k_i — его кривизна, которая для отрезка равна нулю, а для окружности является величиной, обратной к радиусу. Все числа — вещественные, заданные с максимально возможной точностью, и не превосходят 100 по модулю.

Output

Для каждого из наборов выведите, может ли он описывать разбитое круглое зеркало, как показано в примере. Следуйте формату выходного файла максимально точно.

Example

mirror.in
2 2 1 0 0 -1 0 2 2 0.7071067811865475244 0.7071067811865475244 0 -0.7071067811865475244 -0.7071067811865475244 2 2 2 1 0 0 -1 0 2 2 0.7071067811865475244 0 -0.7071067811865475244 0 2
mirror.out
Test case 1 describes a broken round mirror. Test case 2 does not describe a broken round mirror.

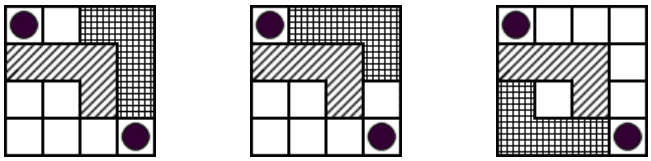
Problem 19. L-Game

Input file:	`lgame.in`
Output file:	`lgame.out`
Time limit:	2 seconds
Memory limit:	64 megabytes

The L-Game was designed by Edward de Bono who enjoys playing games and yet hates to concentrate on a large number of pieces. The intention was to produce the simplest possible game that could be played with a high degree of skill. The L game was the result.

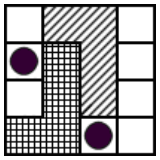
Each player has only one piece, a "L piece". There are also two neutral square pieces. The board is four squares by four squares. The object of the game is to manoeuvre the other player into a position on the board where he cannot move his L piece.

Proceeding from the starting position, the first player (and each player on each move thereafter) must move the L piece first. When moving, a player may slide, turn or pick up and flip the L piece into any open position other than the one it occupied prior to the move. When the L piece has been moved, a player may move either one (but only one) of the neutral square pieces to any open square on the board. It is not required that a neutral piece be moved, this is up to the player!



From any of the three positions above, the player with the checkered L can move his L piece to two new locations (the other two of the three positions above). After moving the L piece, he can move one of the neutral pieces to any of the 6 remaining empty squares, or decide not to move any of the neutral pieces. All in all, there are $2 \cdot (6 + 6 + 1)$ possible moves.

A player wins the game when his opponent cannot move his L piece. In the position below, if the player with the checkered L is to move, he loses, as he can't move his L to a new, unoccupied position on the board.



The game is simple, yet complex as there are so many moves. There are over 18 000 positions for the pieces on the small board and at any moment there may be as many as 195 different moves of which only one is successful.

Write a program that given the position of the pieces and which player is to move, decides if the player has a winning move and output such a move if it exist. If there exists several winning moves, output one of them. If no winning move exists, you should decide whether the game will end in a draw (assuming perfect play from both players), or if the player to move will in fact lose.

A winning move is defined as a move that, no matter how the opponent plays, he cannot avoid losing the game if the player who is about to move continues the game playing perfectly.

A player is losing the game if, no matter which move he plays, he cannot avoid losing if the opponent continues to play the game perfectly.

If neither of these conditions hold (that is, none of the players can force a win), we say the game position is a draw.

Input

The input consist of four lines describing the position of a game in progress. A dot ('.') represents an empty square, an 'x' represents a neutral square piece, '#' represents the L piece of the player who is about to move and '*' represents the L piece of the other player. You may assume that the position is legal and that the player to move has at least one legal move from the current position.

Output

If a winning move exist, output the position after the winning move, using the same format as the input. Otherwise output "No winning move exists" on a line by itself, and on the next line either "Draw" if the game will end in a draw (assuming perfect play) or "Losing" if the player to move will lose the game.

Example

lgame.in	lgame.out
.*** #*.x ###. x...	.*** x*#x ###. ....
...x ###. #*** x..*	No winning move exists Draw
.### x#*x ***.	No winning move exists Losing

Problem 23. Shortest Paths

Input file:	`shortest.in`
Output file:	`shortest.out`
Time limit:	6 seconds
Memory limit:	64 megabytes

You are given a graph with one vertex marked as source s and one as destination t. Each edge of the graph has a positive length. Find the shortest path from s to t. Then find the second-shortest path (the shortest one of all the paths from s to t except the one you've just found). Then the third-shortest, and so on. Output the lengths of first k such paths.

Note that these paths may not be simple, i.e. they may contain some vertex or edge several times (see the 2nd example).

Input

The first line of the input file contains n, the number of vertices of the graph, m, the number of edges of the graph, and k, the number of paths sought ($2 \le n \le 10000$, $2 \le m \le 50000$, $2 \le k \le 10000$).

The second line of the input file contains s and t (integers between 1 and n, inclusive, $s \ne t$).

The next m lines contain the descriptions of the edges, each description consisting of three integer numbers: a b c, denoting the edge from a to b with length c ($1 \le a, b \le n$, $a \ne b$, $1 \le c \le 1000$). There may be more than one edge for the same a and b.

Output

Output k integer numbers in non-decreasing order — the lengths of the paths. In case there are less than k different paths from s to t, output `NO` instead of the lengths of all non-existent paths.

Example

shortest.in	shortest.out
4 5 5 1 4 1 2 1 2 3 1 3 4 1 1 3 1 2 4 1	2 2 3 NO NO
4 4 5 1 4 1 2 10 2 3 10 3 4 10 3 2 10	30 50 70 90 110
2 2 10 1 2 1 2 5 2 1 7	5 17 29 41 53 65 77 89 101 113